Fitts' Law in the Wild: A Field Study of Aimed Movements

Olivier Chapuis, Renaud Blanch, Michel Beaudouin-Lafon

Fitts' Law in the Wild: A Field Study of Aimed Movements

Olivier Chapuis
LRI - Univ. Paris-Sud & CNRS, INRIA
Orsay, France
chapuis@lri.fr

Renaud Blanch
LIG - Univ. Joseph Fourier
Grenoble, France
blanch@imag.fr

Michel Beaudouin-Lafon
LRI - Univ. Paris-Sud & CNRS, INRIA
Orsay, France
mbl@lri.fr

ABSTRACT
This paper presents the first field study of aimed movements in graphical user interfaces, designed to get better insight into pointing in the real (electronic) world and assess the validity of Fitts' law in the wild. We unobtrusively collected kinematic data from 24 users over several months, and segmented it into a table of 2 million movements. We show that Fitts' law is indeed robust for modeling pointing performance, provided that an adequate noise reduction process is applied. We introduce the length-distance index (*LDI*) to take into account the fact that many movements are not straight, and we introduce an extension of Fitts' law that includes an *LDI* term. We also show evidence of the effect of cognitive tasks in pause and click time, and the sensitivity of differences in performance, e.g., across input devices, to *LDI*. Altogether, these findings provide a firm ground to better assess the validity of results obtained in the laboratory.

ACM Classification Keywords
H. Information Systems H.5 Information Interfaces and Presentation H.5.2 User Interfaces (H.1.2, I.3.6)

Author Keywords
Fitts' law, field study, pointing performance, UI logs

INTRODUCTION AND RELATED WORK

Target acquisition using a pointing device is one of the fundamental tasks in graphical user interfaces (GUIs). Since Card et al.'s seminal work [4], a large body of research has been dedicated to improving pointing performance through new devices and/or interaction techniques. At the heart of this research program lies Fitts' law [8]. Fitts' pointing paradigm reduces target acquisition to a unidimensional task characterized by two independent variables: target width (W) and target distance (D). Fitts' law[1] predicts that the movement time (MT) to acquire a target is a linear function of the task's index of difficulty (ID):

$$MT = a + b \times ID, \quad \text{where} \quad ID = \log_2\left(\frac{D}{W} + 1\right). \qquad (1)$$

The index of difficulty fully characterizes the task: higher values of ID, i.e. longer distances and/or smaller targets, require more time and are therefore "harder". The intercept (a) and the slope (b), on the other hand, depend on the context of execution of the task, i.e. the user, the input device, etc.

Fitts' law is the most successful quantitative law used in HCI [21, 26] and has proved extremely robust across a wide range of conditions. It has been extended, among other things, to 2D pointing [22, 10, 11], 3D pointing [9] and multiscale navigation [12]. It is used to compare input devices, e.g., [23], pointing techniques, e.g., [3] and types of users, e.g., [13]. It is even the basis for a standardized experiment used to define the throughput of input devices by the International Organization for Standardization (ISO) [18].

Fitts' pointing paradigm focuses exclusively on the motor control aspect of pointing. Users are instructed to execute aimed movements as fast as possible, with a single target prominently displayed. As a result, Fitts' law models pointing in a GUI in the ideal situation where the user knows exactly where the target is and points directly at it. This is hardly a typical situation: pointing "in the wild" [16] involves other factors than simply the size and position of the target, such as deciding what is the target, dealing with interference from the environment or planning for higher-level tasks. Since so much of the HCI literature depends on the results of controlled experiments based on Fitts' pointing paradigm, it is legitimate to ask if we can expect the results of such experiments to transfer to actual use in everyday GUIs.

The purpose of the work reported in this paper is to test the ecological validity of Fitts' law and more generally to study and analyze aimed movements in the field as opposed to in the laboratory. To our knowledge, it is the first field study of pointing based on a long term collection of pointing data. While field studies are common in HCI and tools exist to collect user data, e.g., for usability studies [19], we are not aware of any field study of the motor aspects of GUIs. For example, Hurst et al. [14] log mouse data to distinguish between novice and skilled users, but their goal is to dynamically adapt the software. Hutchings et al. [15] log window management activity to study the use of multiple monitors, but focus on higher-level tasks.

[1]The formulation in Equation 1 is not Fitts' original one but MacKenzie's widely adopted Shannon formulation [21].

LRI Technical Repport Number 1480
DECEMBER 2007

LABORATOIRE DE RECHERCHE EN INFORMATIQUE
UMR 8623, CNRS Université de Paris Sud
Bât. 490, 91405 ORSAY Cedex France

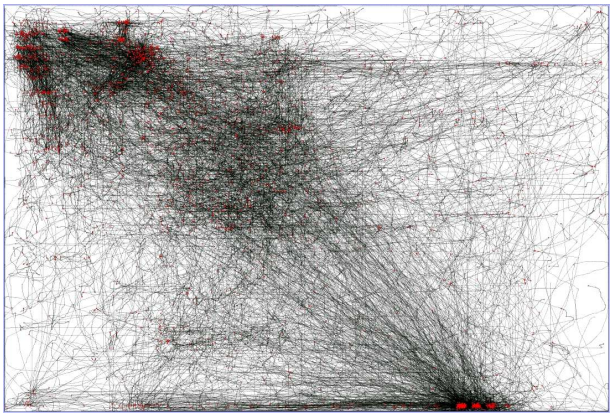

Figure 1. Mouse trajectories (black) and clicks (red).

To conduct the study presented in this paper, we have logged the activity of 24 users over several months with two software probes and analyzed the collected data, which represents over 2 million aimed movements and about 1 billion pixels covered by the cursor (about 219 miles or 352 km at 72 dots per inch). The remainder of this paper describes the methodology we have used to collect and segment the data. Then we proceed with the analysis itself. We introduce the length-distance index (*LDI*) to take into account the fact that many movements are not straight. We show that Fitts' law is indeed robust for modeling pointing performance. We then find evidence that the cognitive part of pointing tasks affect performance and correlate this degradation with *LDI*. This leads to a generalization of Fitts' law that includes an *LDI* term. Finally, we compare performance across input devices and widget types and conclude with some future work.

SYSTEM INSTRUMENTATION

We have developed two software probes targeting two different desktop environments: the X Window System under Linux (X Window) and Apple's Mac OS X (OS X). The probes are unobtrusive, i.e. they run in the background and do not require modifying any of the applications run by the user. The information collected is a kinematic log similar to that of VibeLog [15]: the successive on-screen positions of the cursor with their timestamp, the state changes of the pointing device buttons, and the position, size and stacking order of the windows on screen. We also log the geometry of the widget under the cursor at the time of each click, however for X Window this information is not always available due to technical limitations.

X Window Probe

On X Window we used *wmtrace*, a freely available tool developed to monitor window management activities [5]. *wmtrace* logs pointer events by monitoring the X Window protocol and window information by monitoring the window management protocols: the latter include window decorations (title bar, window buttons, window borders), the buttons in the taskbar and in the pagers. At the beginning of our study (early 2005), it was only possible to log the geometry and type of widget under the cursor for GTK applications.

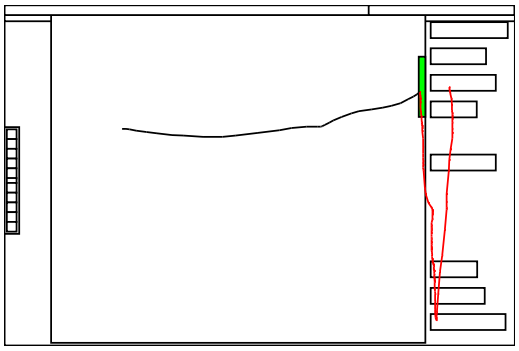

Figure 2. Data recorded by the OS X probe: configuration of the screen; aimed movement (black); target (green); drag movement (red).

We use *wmtrace-view* to replay the logs and display the window frames (without their content) and cursor movements. Logs can be played back and forth as with a video player, and a filtering system similar to DIVA [20] allows to jump, e.g., right before the next click on a close button. We have also developed tools to visualize cursor movements and clicks (Figure 1) and to extract various data tables from the logs.

Mac OS X Probe

On OS X, we have developed a probe that tracks the pointing device events using Apple's IOKit framework to access the low-level USB-HID drivers. The windows' geometry is collected when a button's state changes using the accessibility API provided by the ApplicationServices/HIServices framework. Additional information such as stacking order which is unfortunately not available through the accessibility API is collected using an undocumented functionality of the ApplicationServices/CoreGraphics framework.

The geometry and type of the widget under the cursor is also collected using the accessibility API when a button is pressed. Sixty-nine widget "roles" (*Button*, *MenuBar*, *MenuItem*, *DockItem* etc.) and eight "subroles" (*MinimizeButton*, *ToolbarButton*, etc.) are defined by the API, allowing us to study subsets of the collected data, e.g., all the clicks on the close button of a window. The collected data is sufficient to reconstruct the complete cursor trajectory and the state of the screen each time a button is pressed (Figure 2).

MOVEMENT SEGMENTATION

In order to study aimed movements in the real world, we have to extract such movements from the kinematic data we have collected, i.e. to segment the data into candidate movements and select those that correspond to a pointing task. This step is critical since the quality of the segmentation will determine the validity of our future analyses. In particular, we must be careful and avoid using hypotheses that underlie our analyses, e.g., that aimed movements follow Fitts' law, to segment the source data.

An aimed movement begins when the user decides to reach a target. This cognitive process does not leave an explicit trace in the data we collect, so we need heuristics to identifiy the beginning of a movement. The end of an aimed movement in GUIs almost always corresponds to an explicit

selection action, i.e. a button press (or a button release in the case of a drag). Some movements may not have such an explicit marker, e.g., when hovering over an item to see its tooltip. In the present study, we have decided to focus on aimed movements that end with a button press (↓). We exclude drags because they often correspond to non-aimed movements such as scrolling or inking, and we exclude hovering because we do not have a good heuristic to distinguish it from pauses. The following sections describe in more detail the segmentation process.

Beginning of Movement

To our knowledge, the literature does not address the identification of the beginning of an aimed movement: experimental protocols make the beginning of the movement explicit, e.g., as the first movement after the display of the target. In our case, given a button press at time $t_{\downarrow}$, we know that the movement does not start before the button was last released $t_{\uparrow}$. The movement may have started after $t_{\uparrow}$ however, for example if the user has moved the cursor to help with reading or in case of external interruptions. Therefore we need to analyze the cursor trajectory between $t_{\uparrow}$ and $t_{\downarrow}$.

Points where the cursor velocity is zero, called *pauses*[2], are of particular interest to segment the trajectory. Defining the beginning of the movement as the last pause in the trajectory does not work for two reasons. First, aimed movements consist of one or several submovements [24], which may be separated by (short) pauses. So a pause could be a submovement transition. Second, we have noticed in our data as well as in direct observations that users make long pauses shortly before $t_{\downarrow}$, including immediately before $t_{\downarrow}$, as if they were mentally confirming their action before clicking.

This leads us to use two parameters for the segmentation: the *stop time* and the *verification distance*. Let us call $p_{\downarrow}$ the position of the cursor at $t_{\downarrow}$ and p the position of the cursor at a pause. We define a *stop* as a pause longer than *stop time* that occurs at a distance $d(p, p_{\downarrow})$ greater than the *verification distance*. The beginning of the movement is then the last *stop* before $t_{\downarrow}$. Figure 3 illustrates a cursor trajectory with time running horizontally and the distance along the trajectory running vertically. There is one stop, P_1, one pause in the middle of the movement, P_2, and two pauses close to $t_{\downarrow}$, P_3 and P_4. Since P_1 is the last stop before $t_{\downarrow}$, the beginning of the movement is the end of that pause (t_b). We call *movement time* (MT) the time between t_b and $t_{\downarrow}$, *verification time* (VT) the total duration of the pauses that occur within the verification distance, and *click time* (CT) the duration of the pause, if any, that occurs immediately before $t_{\downarrow}$. Finally, we call *pause time* (PT) the total duration of the pauses that occur during the movement excluding those within the verification distance, i.e., in this example, P_2.

Given these definitions, we need to pick adequate values for the two parameters of the segmentation, *stop time* and *verification distance*. Hwang et al. [17] have studied the mouse movements of motion-impaired users with able-bodied users as control. This study gives some statistics about "pauses" between submovements for able-bodied users. Although no precise definition of a pause is given, the mean number of pauses in a movement was about 1 and the mean time of a pause was about 70 ms. Using the kinematic data collected for a previous Fitts experiment, we determined that with a *verification distance* of 10 pixels, the mean pause time (PT) was 75 ms and 95% of the pauses were less than or equal to 240 ms. The mean verification time (VT) was 185 ms, with 592 ms for the 95% quantile.

[2]A pause is a period of 50 ms or more with no cursor motion. This delay is needed to account for the irregular sampling of the cursor.

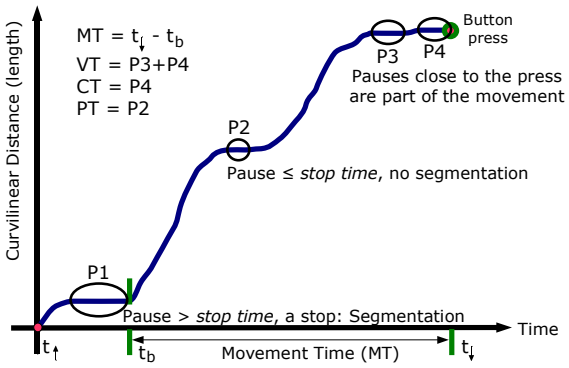

Figure 3. Segmentation of a cursor trajectory

For the study presented in this paper, we chose a *verification distance* of 10 pixels and a *stop time* of 300 ms. As the stop time seemed quite long, we used our replay tool to make sure that it was a reasonable value through extensive observation of the data. Note that the verification distance means that we missed movements whose amplitude is less than 10 pixels. We believe this is acceptable since such small movements are of limited interest. Also, since we have no restriction on VT, we may have extremely long movements where the click time largely dominates the movement time.

End of Movement

As stated earlier, we only analyze movements that end with a button press, making the segmentation of the movement's end trivial. In order to distinguish among different types of final actions at the end of the movement, we measure the duration (t_{drag}) and distance traversed (d_{drag}) while the button is pressed, i.e. between $t_{\downarrow}$ and the following $t_{\uparrow}$. Using an analysis of the distributions of d_{drag} and t_{drag}, we determined thresholds to distinguish among the following final actions: a drag ($d_{drag} > 5$ pixels) or a click ($d_{drag} \leq 5$ pixels). Clicks are further subdivided into two categories: regular clicks ($t_{drag} \leq 300$ ms), and long clicks ($t_{drag} > 300$ ms). Finally, if the difference between $t_{\uparrow}$ and the following $t_{\downarrow}$ is less than or equal to 250 ms, that movement is discarded and the previous final action is concatenated with the current one as a combination, such as a double click.

Since we plan to conduct a Fitts' analysis of the aimed movements that have been segmented, we need to know the *geometry* of the aimed target. We call *targeted movements* those movements for which we have reliable information about the target, i.e. movements for which we can be fairly confident

that the widget under the cursor was the actual target. Our data logs capture information about the widget under the cursor. We use the widget type to classify the movement as a targeted movement. For example, if the widget is a text area, the actual target is likely to be a character in the text as opposed to the whole text area. Since we do not have information about the actual target in this case, we do not record the movement as being targeted and eligible for a Fitts' analysis.

DATA SUMMARY

We now present the data we have collected and a first analysis of the results of the segmentation. We developed our own software for the segmentation and thoroughly verified the results. Analyses were conducted with the R[3] and JMP[4] software packages. After segmentation, our main data table contains over 2 million rows (one per segmented movement).

Users Characteristics

Twenty-four users participated in the study, 20 men and 4 women, all right-handed. All users were either students, colleagues or engineers from different institutions and were expert computer users. Ten participants used their computers in two configurations, one in three configurations. A configuration is a combination of computer, desktop environment (for Linux users), pointing device (mouse or trackpad) and number of screens. For example, a user may switch to another environment under Linux; a laptop user may use a second monitor and a mouse when working in his office. This resulted in 36 different configurations, listed on the left of Table 1: 9 participants and 13 configurations under X Window, 15 participants and 23 configurations under OS X.

Participants used the probing software from several weeks to several months, and we regularly collected the log files. The second part of Table 1 summarizes the results of the segmentation to give an idea of the magnitude of the data (the total duration of logged data is not included because it would include long periods of time where users are away from their computer): total number of movements (mvts), sum of the straight distances between the beginning and end of each movement (distance), total distance covered by the cursor (length), length-distance ratio (*LD*) described below. The last section of the table lists the same data for targeted movements, i.e. movements for which we know the target. These represent 22% of all movements. The last column (mean Fitts equation) will be explained later.

Analysis of Final Actions

Each movement ends with a final action such as a click, a double click or a long click. Most final actions ($96.89 \pm 3.39\%$) are carried out with the left button[5]. The variability across configurations stems from the differences among desktop environments. On OS X, most participants use a single-button trackpad or mouse, or have not configured the right and middle buttons. Only one OS X user uses the right button more than 1% of all clicks (3.07%). A few OS X users use the middle button, with a maximum of 4.7% for a user of Exposé, a window management function used to navigate among windows. On X Window, the right button is used in $4.43 \pm 2.41\%$ of all clicks and the middle button in only $1.76 \pm 1.56\%$ of all clicks. The middle button pastes the primary selection and all of the X Window users know this feature, with a maximum of 4.64% of all clicks.

A large majority of left button actions are single clicks, then drags, then double clicks and then long clicks. On OS X we have $77.06 \pm 8.02\%$ clicks, $15.54 \pm 6.55\%$ drags, $4.49 \pm 2.53\%$ double clicks, $1.42 \pm 1.29\%$ long clicks and less than 2% other combinations, e.g., triple clicks. On X Window we have $72.56 \pm 6.04\%$ clicks, $17.75 \pm 4.66\%$ drags, $4.71 \pm 1.84\%$ double clicks, $2.2 \pm 1.29\%$ long clicks and $2.75 \pm 1.7\%$ other combinations.

The Length Distance Ratio and Length Distance Index

One of the consequences of pointing in 2D is that the movement trajectory is not fully determined by the specification of the target as in a traditional 1D Fitts' experiment. While a straight line may not be the fastest way to reach a target because of the motor control involved in pointing, it is a good approximation. To measure the deviation from this ideal trajectory, we use the *length-distance ratio* (*LD*), computed as the ratio of length of the movement to direct distance between the beginning and end points of the movement.

We observed that *LD* could be huge. We have 701 movements with $LD > 100$ and 1.42% of the targeted movements with $LD > 10$. These large values are mostly due to movements with short distances: all but 8 movements with $LD > 100$ have a distance $D < 80$, and 75% of the movements with $LD > 10$ have a distance $D \leq 100$. Obviously, it is easier to obtain a large *LD* ratio when *D* is small.

Analysis of movements with large values of *LD* using the *wmtrace-view* tool revealed patterns such as large circular, spiral or zig-zag movements. These could correspond to a user showing something on the screen, or to an attempt at locating the cursor after having lost it (particularly on a large screen or 2-monitor configuration) by moving it and then proceeding with a standard aimed movement.

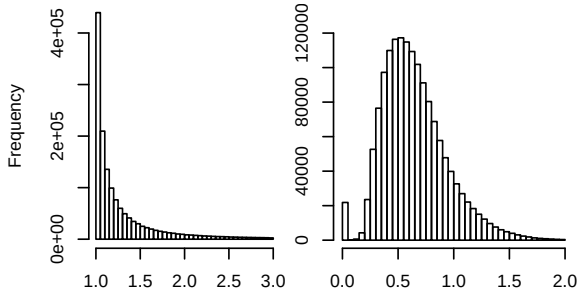

Figure 4. *LD* **(left) and** *LDI* **(right) distributions** ($D \geq 100$) **.**

Table 1 lists the mean values of *LD* per configuration. Figure 4 (left) shows the distribution of *LD* across all aimed

[3] http://www.R-project.org

[4] Version 6. SAS Institute Inc., http://www.jmp.com/

[5] We use the notation $m \pm \sigma$ where m is the mean and σ the standard deviation of the value across configurations. Unless otherwise noted, we first take the mean for each configuration and then compute the mean and deviation across configurations.

Configuration					Aimed movements				Targeted movements				
#	user	env	mouse	scr	mvts	distance	length	LD	mvts	distance	length	LD	mean Fitts Equation (100-perc.)
1	1	X (kde)	mouse	1	32967	10683768	15934682	1.49	5747	3049126	4885954	1.6	$812+182ID$, $r^2=0.933$
2	2	X (gnome)	mouse	1	13544	4282891	5461624	1.28	4223	1293721	1640946	1.27	$155+236ID$, $r^2=0.908$
3	2	X (kde)	mouse	1	163369	45574797	58148641	1.28	29718	10700182	14309843	1.34	$301+232ID$, $r^2=0.948$
4	3	X (gnome)	mouse	1	167030	48664264	65893665	1.35	16762	7889304	10537697	1.34	$723+112ID$, $r^2=0.838$
5	4	X (fvwm)	mouse	1	36101	11195957	18840848	1.68	5877	2256248	3841300	1.70	$575+164ID$, $r^2=0.967$
6	4	X (gnome)	mouse	1	33708	9862120	17023748	1.73	4513	1854254	2945529	1.59	$571+176ID$, $r^2=0.968$
7	5	X (gnome)	mouse	1	119365	40725024	61681858	1.51	20205	8666703	12796705	1.48	$690+159ID$, $r^2=0.945$
8	5	X (gnome)	mouse	2	225162	72537849	112624577	1.55	11205	6135442	9255788	1.51	$538+185ID$, $r^2=0.962$
9	6	X (icewm)	mouse	2	123972	40177199	62753069	1.56	4977	2735721	4159078	1.52	$716+179ID$, $r^2=0.929$
10	7	X (fvwm)	trackpad	1	294703	86118404	110425445	1.28	51790	19171145	25180623	1.31	$1041+168ID$, $r^2=0.929$
11	8	X (gnome)	mouse	1	59836	16233819	21389240	1.32	1742	653191	834999	1.28	$551+127ID$, $r^2=0.950$
12	9	X (gnome)	mouse	2	90070	33051271	54114877	1.64	6647	3796564	5935634	1.56	$531+203ID$, $r^2=0.963$
13	9	X (xfce)	mouse	1	39592	11251573	18759402	1.67	1794	811253	1466646	1.81	$363+220ID$, $r^2=0.918$
14	10	osx	mouse	2	24554	7687145	10550344	1.37	9615	3651996	5019558	1.37	$647+191ID$, $r^2=0.955$
15	10	osx	NA	1	63660	18343411	25749711	1.40	25534	8402070	11670058	1.39	$749+184ID$, $r^2=0.953$
16	11	osx	mouse	2	13099	4456521	6245852	1.40	3136	1269682	1875338	1.48	$509+166ID$, $r^2=0.954$
17	12	osx	mouse	1	142874	52756643	74527356	1.41	63559	22740180	32659908	1.44	$571+178ID$, $r^2=0.950$
18	12	osx	trackpad	1	75186	19495234	25680509	1.32	28036	7254829	9777142	1.35	$775+164ID$, $r^2=0.954$
19	13	osx	mouse	1	18908	6177282	8562112	1.39	8746	3010629	4168809	1.38	$765+209ID$, $r^2=0.978$
20	13	osx	mouse	2	23001	7767334	11057052	1.42	12411	4128589	5933798	1.44	$786+194ID$, $r^2=0.967$
21	13	osx	trackpad	2	6440	2418337	3263012	1.35	3016	1143875	1557191	1.36	$760+207ID$, $r^2=0.978$
22	14	osx	mouse	2	1613	478595	754812	1.58	885	267756	415011	1.55	$505+222ID$, $r^2=0.969$
23	14	osx	NA	1	3589	831223	1297690	1.56	2074	490374	768597	1.57	$815+156ID$, $r^2=0.988$
24	15	osx	mouse	1	18173	7026508	8380156	1.19	3402	1201124	1493224	1.24	$541+119ID$, $r^2=0.906$
25	15	osx	NA	1	12499	3331979	4127239	1.24	5613	1438667	1801102	1.25	$575+132ID$, $r^2=0.929$
26	16	osx	mouse	1	66031	24725760	30360697	1.23	40029	15764719	19503241	1.24	$504+144ID$, $r^2=0.959$
27	17	osx	mouse	1	53782	16235427	27511387	1.69	22431	7330223	11960445	1.63	$601+191ID$, $r^2=0.948$
28	18	osx	mouse	1	45124	14996810	19902184	1.33	20860	8029698	10643716	1.33	$640+154ID$, $r^2=0.902$
29	18	osx	NA	1	15692	4863893	6331967	1.30	9696	3252079	4232195	1.30	$701+192ID$, $r^2=0.955$
30	19	osx	trackpad	1	58091	16156515	22373433	1.38	13646	4543651	6313030	1.39	$524+144ID$, $r^2=0.965$
31	20	osx	NA	1	2141	289900	370801	1.28	934	132188	167946	1.27	$735+152ID$, $r^2=0.976$
32	21	osx	mouse	1	11094	3602669	4997080	1.39	7036	2365282	3308390	1.40	$469+171ID$, $r^2=0.920$
33	22	osx	NA	1	22255	6174178	8223251	1.33	8290	2370337	3197145	1.35	$742+120ID$, $r^2=0.740$
34	23	osx	mouse	1	10464	3727996	6176327	1.66	3390	1337742	2200392	1.64	$903+193ID$, $r^2=0.947$
35	24	osx	mouse	1	10447	4470889	5867890	1.31	3716	1538955	2067458	1.34	$584+159ID$, $r^2=0.948$
36	24	osx	mouse	2	31387	12542161	17265467	1.38	9203	3827855	5395187	1.41	$606+176ID$, $r^2=0.967$
total / mean$_{\pm stddev}$					2129523	668915357	952628016	$1.42_{\pm 0.15}$	470458	174505365	243919633	$1.43_{\pm 0.14}$	$627_{\pm 168}+174_{\pm 31}ID$, $0.941_{\pm 0.044}$

Table 1. Summary of participants, configurations and collected data. Aimed movements are all the movements resulting from the segmentation. Targeted movements are those for which we know the geometry and type of the target.

movements with $D \geq 100$. It is highly skewed and looks like a power law. Looking at the quantiles, we find that the mean (across configurations and $D \geq 100$) of the medians is 1.15 ± 0.06. The mean of the first quartile is 1.05 ± 0.02 and the mean of the third quartile is 1.48 ± 0.18. The difference between OS X and X Window is minor and we obtain similar values when considering only targeted movements. Thus, while a large fraction of the movements are almost straight, a significant number are not. For example, 10% of movements with $D \geq 100$ have $LD \geq 2.46$. We compared this data to that from a previous controlled Fitts experiment and found the mean value of LD to be 1.045 ± 0.09 (median $= 1.025$) with 90% of the movements having $LD \leq 1.178$.

The LD distribution looks like a power law, has a very long tail and strongly depends on distance. It is thus difficult to interpret. Instead, we propose to consider the *length distance index* (*LDI*) defined as follows:

$$LDI = (\frac{L}{D} - 1)^{\frac{1}{4}}$$

This transform of LD breaks the power law, normalizes the small LD values and concentrates large LD values. Figure 4 (right) shows the resulting plot. The mean LDI is 0.66 ± 0.06 and the median, close to the mean, is 0.61 ± 0.07, matching the peak of the distribution. As we will see later, this index will prove useful in our analyses.

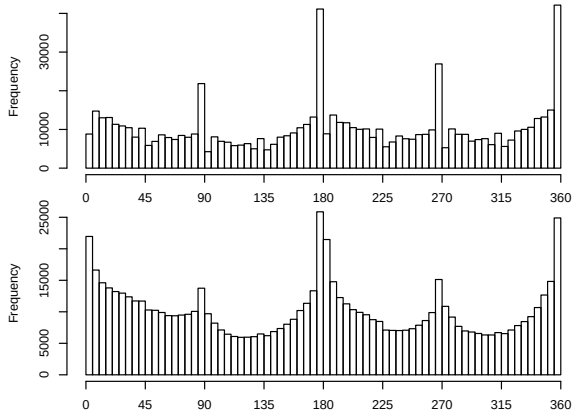

Figure 5. Distribution of movement direction at the beginning of the movement (top) and from end to end (bottom).

Variability in Movement Direction

Figure 5 plots two distributions of movement directions: the direction at the beginning of the movement (top) and the overall direction of the movement, i.e. the direction of the straight line going through the beginning and end points of the movement (bottom). While the aliasing due to the discretisation of cursor positions may explain the privileged directions at the beginning of the movement (0, 90, 180, 270°), we do not explain the privileged directions (horizontal and vertical) for the full movement.

We find similar distributions when considering only targeted movements and/or when removing movements with small distances. However, when analyzing movements by widget type, we find that some widget types elicit movements with horizontal, vertical and sometimes diagonal directions because of their position on the screen (e.g., OS X Menu Bar items or TaskBar buttons), the geometry of the widget (e.g., large and thin TitleBar vs. small TitleBarButton) or their position relative to the cursor (e.g., popup menu items).

Pointing Task Variables

By studying targeted movements only, i.e. those movements for which we know the geometry of the target, we can study the characteristics of the underlying pointing task, i.e. the independent and dependent variables of Fitts' law, with the notable exception of error rate, since by definition all of our movements are hits.

Since we are studying a 2D pointing task, we must decide what distance, width and ID to consider. There are a number of generalizations of Fitts' law to 2D in the literature, including MacKenzie & Buxton [22], Accot & Zhai's bivariate pointing [1] and Grossman & Balakrishnan's probabilistic model [10]. While the latter was not applicable to our data since it requires hits and misses we tried several of the other formulas and found little differences among them. Therefore we opted for the simplest model, MacKenzie & Buxton's minimal size: the size of the target is W_{min}, the smallest dimension of the target, and the index of difficulty is ID_{min}, computed from Equation 1 with W_{min} instead of W.

Figure 6 shows the distribution of distance, W_{min} and ID_{min} for the targeted movements of our data set. Note the two peaks around $ID = 1$ and $ID = 5$. Most of our targets have a W_{min} of about 20 pixels, corresponding to buttons, title bars, scrollbars, etc. There are probably more targeted movements with higher ID_{min} in real world pointing but they are not accessible to our logging tools (we have no such targets with our X Window probe and with OS X it is difficult to know the real W_{min} for widgets such as the text area).

Figure 6 also shows the distribution of click time (CT), verification time (VT) and movement time (MT). The distributions of CT and VT are very similar. Indeed, CT contribute 80% of VT. More importantly, we note that CT is quite large: after removing outliers by considering movements with a movement time of 5 sec at most, the mean click time CT is 277 ± 78 ms and 20% of these targeted movements have $CT \geq 380$ ms. This was confirmed by examining some of the data with *wmtrace-viewer*. We will come back to this observation later in the paper.

FITTING FITTS' LAW

In this section, we investigate the role of ID on dependent variables. We especially check its relationship to movement time (MT), i.e. whether Fitts' law holds for our data. The data set is the set of targeted movements, for which we have target information.

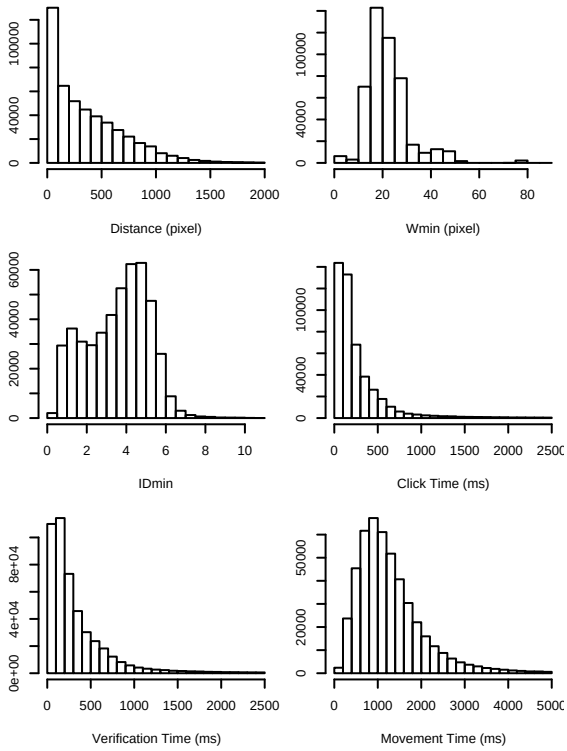

Figure 6. Distributions of the dependent and independent variables of targeted movements.

Mean Fit

In a controlled experiment, continuous factors such as ID are typically sampled over a fixed range. The dependent variables are then averaged for each factor value and those means are fitted. In a typical Fitts' experiment, this procedure leads to very good linear fit of MT vs. ID, i.e. an adjusted r^2 close to 1.

As in our case ID is continuous, we have to partition the observations along the ID axis before averaging MT. ID is thus partitioned in q-quantiles (q intervals each containing the same number of observations). The linear correlation between the means of the factor and the dependent variable over each quantile can then be computed. The process is further refined by removing outliers: we restrict our analysis to targeted movements that are less than 5 seconds long, representing 98.90% of all targeted movements (470,458 observations). Outliers typically correspond to long or multiple pauses during the movement (PT) or at the end (CT). Figure 7 shows the result of this process for one user.

Without averaging, the model has a poor fit: $MT = 690 + 163 \cdot ID$ ($r^2 = 0.117$). With an averaging over 10-quantiles, the model is $MT = 679 + 166 \cdot ID$ ($r^2 = 0.992$) , and $MT = 686 + 164 \cdot ID$ ($r^2 = 0.978$) with a 100-quantiles decomposition. Finer decompositions lead to a degradation of r^2 ($r^2 = 0.932$ for 1000 quantiles; $r^2 = 0.727$ for 10000 quantiles) due to each quantile containing fewer samples (less than 50 points in the last case). The same phenomenon occurs when the data is fitted separately for each user. Table 1

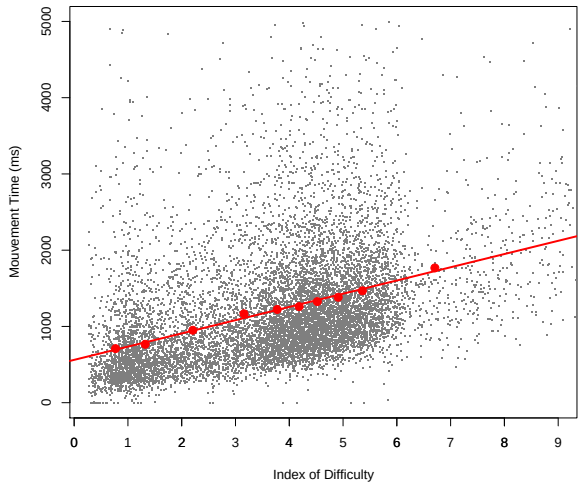

Figure 7. Fitts' linear fit of MT by ID using the averaging process for one user (# 4). Small (black) points represent all the points. Large (red) points are the means of each quantile.

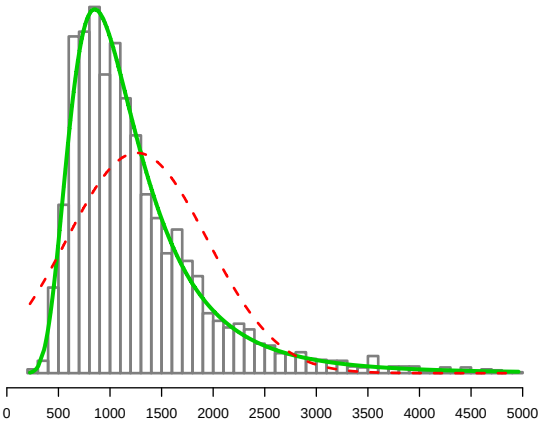

Figure 8. Typical distribution of movement time for a percentile interval ($3.6 \leq ID_{min} \leq 3.85$, 3179 values) for configuration # 17. The dashed (red) line is the normal fit (mean 1271, deviation 696) and the plain (green) one is the *gev* fit (location 931, scale 391 and shape 0.24).

(last column) lists the individual fits using a q-quantiles decomposition such that q is the largest value with at least 200 observations per quantile. The average individual model is $MT = 627_{\pm 168} + 174_{\pm 31} \cdot ID$ ($r^2 = 0.940 \pm 0.044$).

Effects of the 2D Index of Difficulty

The previous statistics were computed with the ID_{min} formulation of ID [22]. We compared standard errors on the 100-quantiles partition with various other formulas for a 2D ID. $ID_{W'}$, which uses the apparent width (the extent of the target intercepted by the movement support) [22] and $ID_{p,\omega,\eta}$, Accot & Zhai's bivariate pointing model [1], did not result in better fits than ID_{min}. We also tried to use L, the length of the movement, instead of D in the ID formulas, but it led to worse r^2 values despite the fact that L is a dependent variable directly linked to the movement execution. We also tried $ID = D/W_{min}$ ($r^2 = 0.414$) and $ID = \sqrt{D/W_{min}}$ ($r^2 = 0.839$) as in Meyer et al. [24]. The second best formulation was in fact Fitts' original formulation with W_{min} for the target extent: $ID = \log_2(2D/W_{min})$ leads to $r^2 = 0.966$!

Alternative Fits

Figure 8 shows a typical distribution of movement time for a q-quantile. Since these distributions are skewed, the mean is not a good representation of the values. The generalized extreme value (*gev*) distributions[6] provide a better match, and we can use the *location* parameter (the position of the maximum) instead of the mean to represent MT [6]. Using this definition of MT, the linear fit is even better: $MT = 346 + 167 \cdot ID$, $r^2 = 0.991$. This method significantly changes the intercept of the model but the slope is very close to that of the mean fit. Also, the *gev shape* parameter (characterizing the skewness) decreases linearly with ID, from ≈ 0.4 to ≈ 0 while the *scale* parameter (characterizing the width of the distribution) increases linearly from ≈ 300 to ≈ 600. This means that as ID increases, the distributions are less skewed and wider, getting closer to a normal distribution.

Another way to handle the non-normality of the MT distribution is to study the quantiles for this axis. Figure 9 shows the evolution of Fitts' law parameters and r^2 values when only the $p\%$ fastest movements are considered for each percentile of the ID distribution. The fits are very good from the 10th percentile to the 95th percentile, indicating that the law is robust. As we consider slower and slower movements, the slope and intercept increase, showing the performance degradation. Beyond the 95th percentile, the fit collapses, meaning that Fitts' law no longer applies.

Fitting Restricted Data Sets

The analyses of the previous sections lead to similar results for each individual configuration (although in some cases, the ends of the curves in Figure 9 are irregular instead of dropping). The same analyses can be conducted on subsets of the data. Table 2 shows Fitts' law parameters for the mean and *gev* fits by input device (mouse vs. touchpad), screen configuration (1 vs. 2 screens) and environment (X Window vs. OS X). It shows, for example, that the mouse and the touchpad have similar slopes but quite different intercepts. Pointing seems slightly faster with a single screen than with a dual screen setup. There is no clear difference between OS X and X Window.

Subset	a mean/gev	b mean/gev	r^2 mean/gev
All	686/359	164/163	0.978/0.971
Touchpad	884/540	162/175	0.959/0.952
Mouse	583/284	174/167	0.987/0.984
2 scr & Mouse	665/372	180/167	0.977/0.978
1 scr & Mouse	572/270	170/164	0.981/0.980
X Win. & Mouse	550/272	180/167	0.964/0.966
OSX & Mouse	603/287	170/167	0.991/0.992
$LD < 1.2$	369/189	171/157	0.977/0.980
& # pause < 3	400/213	147/141	0.982/0.972
& $CT < 300$	243/177	157/141	0.983/0.970

Table 2. Fitts parameters for subsets of the data (mean and *gev* 100-percentile process). For the last three lines, conditions are cumulative.

[6] The gev distributions can be defined as the limit distribution of the minima (or maxima) of a large number of independent random values from a single probability distribution. They are used, in general, to model the minima of long sequences of random variables.

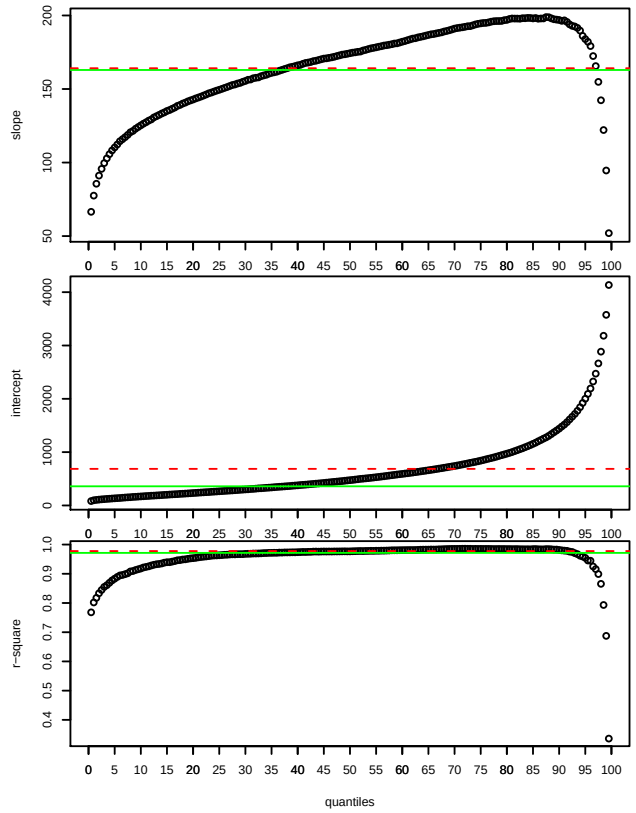

Figure 9. Fitts' law parameters and r^2 for MT quantiles (0.5 % steps). The dashed (red) horizontal line represents the mean fit and the plain (green) horizontal line represents the *gev* fit.

The last 3 lines of Table 2 show what happens when gradually constraining the data set according to movement properties. The first constraint ($LD < 1.2$) selects movements that are almost straight. This substantially reduces the intercept. The second constraint (# pause < 3) further selects movements with no submovements. This reduces the slope. The third constraint ($CT < 300$) further selects movements with short click times. Together, these constraints amount to selecting those movements that are closest to those observed in a controlled experiment where movements are straight, direct and without pause at the end. The slope and intercept then match the typical values reported in the literature for controlled experiments. Overall these analyses show that Fitts' law indeed holds in the wild.

Fitts' Law and the Cognitive Part of a Pointing Task

Using a reduction process, we have seen in the previous sections that Fitts' law can adequately model *in situ* pointing. The difference with values reported for controlled experiments is mainly the magnitude of the intercept: real-world pointing is slower than that observed in the lab. Several factors can explain this difference: first, in everyday pointing users are not instructed to point "as fast as possible"; second, pointing can be disturbed by external factors, such as being interrupted by a phone call or speaking to someone; and finally, pointing in the real world involves cognitive activities that are not related to the motor control itself, such as: (i) making a choice, e.g., when answering a dialog box; (ii) finding an item, e.g., in a menu or in the taskbar; (iii) double checking before a dangerous action , e.g., when closing a window or deleting a file. Such cognitive activities may take place during the movement [4] and are likely to interfere with motor control and therefore pointing performance. While we do not have direct traces of this cognitive activity, we can search for clues in our data.

Observation of movement ends shows that users often spend a fairly long time before clicking once they have stopped in the target. Analysis of variance shows that CT and VT depend on target size and, more surprisingly, on distance (CT and VT decrease when W and D increase). However, these dependencies seem to be weak; we cannot find, and do not think there is any, robust law that can predict CT and VT based on the geometric properties of a pointing task. Since we do not have a motor-control explanation for these long click and verification times, and since they are not observed in traditional Fitts' experiments, we strongly suspect that they are due, at least in part, to cognitive activity during pointing.

Another measure that may indicate cognitive activity during the movement is pause time (PT). We found a very strong correlation between PT and LD, i.e. pause time is longer when movements are not straight. Using the length-distance index LDI instead of LD, we found a very good *linear* correlation between PT and LDI: $PT = 4.753 + 318.2 \cdot LDI$, $r^2 = 0.990$ (10-percentile mean fit for all the data) and $PT = -4.667_{\pm 54.19} + 309.9_{\pm 82.68} \cdot LDI$, $r^2 = 0.919 \pm 0.053$ (10-percentile mean fit by configuration). We verified on subsets of our logs that pauses often correspond to changes in direction of movement, which could explain this correlation. These changes are most likely due to cognitive activity, e.g., changing one's mind while reaching for a target.

Since LDI correlates with longer pauses and therefore longer movement time, we are interested in combining it with ID to create a *combined index of difficulty* (CID) that accounts both for the difficulty of the motor task and the "distraction" caused by cognitive activity. Using an ANOVA for the model $MT \sim ID + LDI + ID \cdot LDI$ with the configuration as random factor, we found, as expected, a strong effect of ID ($F_{1,465275} = 106852$, $p < 0.001$), an even stronger effect of LDI ($F_{1,465276} = 125626$, $p < 0.001$) and an $ID \times LDI$ interaction effect ($F_{1,465263} = 16328$, $p < 0.001$).The interaction between ID and LDI can be easily explained in terms of Fitts' Law: as LDI increases the slope and intercept of the Fitts equation increase as well. This is particularly visible if we use the measure $MT - VT$: Figure 10 shows the successive fit lines when we restrict the data to successive LDI intervals (8-percentile).

Analysing the results of the ANOVA leads to the following definition of CID:

$$CID = ID + 8 \cdot LDI + 4 \cdot ID \cdot LDI$$

which gives the following model: $MT = 362 + 49 \cdot CID$, $r^2 = 0.302$ for the complete data set (vs. $r^2 = 0.117$ when fitting with ID). Since we have shown that CT and VT in-

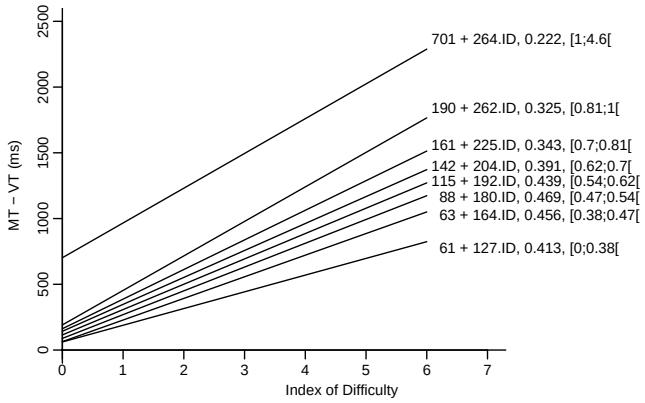

Figure 10. Fit lines for successive *LDI* intervals. Next to each fit line is the Fitts equation, r^2 value and *LDI* interval.

troduce noise that we cannot yet control, we obtain even better fits with $MT - CT$ ($r^2 = 0.497$ vs. 0.207 with *ID*) and $MT - VT$ ($r^2 = 0.541$ vs. 0.231 with *ID*). We obtain similar improvements when restricting the fit to individual users, input device, and configuration. Since straight movements, which are the norm in controlled experiments, have an *LID* close to zero, these results indicate that *CID* is a possible generalization of Fitts' *ID*.

COMPARATIVE PERFORMANCE ANALYSIS

We now turn to a comparative analysis of pointing performance. First we analyze differences among configurations by looking at the effect of desktop environment, input device and number of screens. Then we analyze the effect of the targeted widget's type.

Desktop Environment, input device and Screens

We already noted in our Fitts' analysis a significant difference in performance between mouse and touchpad in favor of the mouse. In order to analyze the effects of both the desktop environment and input device, we performed a full factorial ANOVA of movement time with *ID*, *LDI*, *Input* and *Env* as factors and *User* as a random factor (some users have used both a mouse and touchpad).

We found no significant effect of *Env* on movement time. This may be seen as a proof of the common wisdom that desktop environments have become very similar, as well as a testament to the robustness of our logging tools.

We found strong simple effects of *ID*, *LDI* and *Input*. A post-hoc test reveals a difference in mean of 369 ms between the mouse and touchpad. However, we found an interaction between *Input* and *Env*. For the touchpad, the mean movement time is 533 ms for X Window and 205 ms for OS X but there is no significant difference for the mouse. However, this difference vanishes if we remove the click time (*CT*) and use $MT - CT$ as measure: there is no difference between the X Window user and all the mouse users. On the other hand, we still see a significant difference between the touchpad and the mouse for OS X users. We must be careful however in interpreting these results since there is only one X Window touchpad user.

We further investigate interactions between *ID*, *LDI* and *Input* by restricting our analysis to OS X users. We find $ID \times Input$ and $LDI \times Input$ interaction effects. In both cases, as the index (*ID* or *LDI*) increases, the difference between mouse and touchpad becomes smaller. If we take $MT - CT$ as a measure, we find strong simple effects of *ID*, *LDI* and *Input* and a difference in means of 105 ms. We find no interaction between *ID* and *Input* but an interaction between *LDI* and *Input* that can be explained by the reduction of the difference between mouse and touchpad for large *LDI*.

It is not surprising that, as *LDI* grows, the difference between mouse and touchpad shrinks: as movements are less straight and less efficient, the inherent capabilities of the input device have less impact on performance. It is more surprising however that the same thing happens with *ID*. This is consistent with Yun & Lee [27] but in contradiction with Epps [7] and Amer et al. [2]. A possible interpretation is that OS X has an optimized transfer function for the touchpad. Note also that 50% of the difference is contained in the click time showing the importance of click techniques for touchpads [23].

We found no robust effect when investigating the effect of screen configuration even when considering single-screen movements vs. cross-screen movements. It is clear however that in a dual screen configuration, there is a primary screen where almost all clicks are performed.

Widget Type

We now turn our attention to the effect of widget type. The results presented here are based on mouse movements only, however they are still valid when considering the complete data set, or touchpad users only, or each user separately. We used *ID*, *LDI* and *Widget* as factors and the configuration as a random factor and performed several ANOVAs with different subsets of widget types.

First, we focus on widgets with different shapes: the window title bar (long rectangle) and buttons (square), the window border (thin rectangle) and the resizing grip (small triangle at the bottom right of the window). We test two definitions of Fitts' index difficulty for 2D pointing: ID_{min} [22], which uses the smallest dimension, and $ID_{2,1,1}$ [1], which takes into account the aspect ratio[7].

With $ID = ID_{min}$, we find significant simple effects of *ID*, *LDI* and *Widget* and small $Widget \times ID$ and $Widget \times LDI$ interaction effects caused by the window border becoming faster than the other widgets as *ID* or *LDI* increases. A post-hoc Tukey HSD test shows the following ordering ($>$ means faster with the difference indicated in parentheses): border (42 ± 13 ms) $>$ title bar (74 ± 13 ms) $>$ resizing grip (175 ± 13 ms) $>$ window button. This shows a substantial difference in mean of 248 ± 13 ms between the window title bar and the window buttons.

[7] $ID_{2,1,1} = \log_2(\sqrt{(D/W)^2 + (D/H)^2} + 1)$

With $ID = ID_{2,1,1}$, we find similar simple and interaction effects, but no significant difference in mean between border and resizing grip and between resizing grip and title bar (we still have a small difference between title bar and border: 49 ± 12 ms). There is still a large difference between the title bar and buttons (169 ± 5 ms for a grand mean of 1500 ms). This difference does not depend on *ID* but decreases as *LDI* grows. We believe this difference can only be caused by cognitive activity. Indeed, closing or iconifying a window has more dire consequences than moving or resizing it, in that the cost of undoing the action is higher. This explanation is supported by the fact that the difference is mostly concentrated in the click time *CT* (135 ± 8 ms).

We then test the particular case of widgets that are on the border of the screen, such as the menu bar or the dock icons on OS X: since the cursor is blocked by the border of the screen, these widgets effectively have a semi-infinite size and should be easier to target. We find a significant difference in mean of 146 ± 5 ms. Again, this difference is smaller with larger *LDI* values (down to zero for large *LDI*) and there is a significant but small interaction with *ID* (less that 10 ms difference between $ID = 1$ and $ID = 7$). As before, we observe that the difference in movement time is mostly due to the verification time. Overall, our data supports the hypothesis that targets on the border of the screen improve pointing performance by allowing overshooting.

Another special type of widget are menu items. First, popup menus appear next to the cursor and have low *ID*s, which should improve performance. Second, long menu items often require visual scanning, which should impair performance. We found evidence of these two hypotheses: menu items are one of the fastest widgets for small *ID* and one of the slowest for large *ID*, i.e. we observe a strong interaction of *ID* by *Widget* when considering the menu item widget type against the other types. This supports designs such as the split menu [25].

CONCLUSION

We have described what we believe is the first field study of aimed movements. Using two unobtrusive software probes, we have collected kinematic logs from 24 users in 36 hardware/software configurations over several months. We have described our strategy to segment this data into a collection of over 2 million aimed movements and a subset of about half a million targeted movements, i.e. movements for which we know the geometry and type of the target. The goal of this study was to gain insight into real-life pointing and assess the validity of the results of controlled experiments of pointing techniques in the real world. In particular, we wanted to test the validity of Fitts' law in the wild.

The results of our study can be summarized in the following findings. First, we introduced the length-distance index (*LDI*) as a complementary tool to the index of difficulty (*ID*) in order to better understand aimed movements that are not straight. Second, we showed that Fitts' law was a good predictor of mean performance even when using separate quantiles and *gev* locations, demonstrating the remarkable robustness of the law. This strongly supports the fact that results gathered in the lab with controlled Fitts' experiments can be valid in the field. However, we also measured large intercepts of Fitts' regression lines and found a lot of noise in the data. We used the *LDI* to obtain better fits, first by filtering out movements with large *LDI*, then by introducing a variant of Fitts' law that includes an *LDI* term. Since movements are mostly straight in Fitts' studies (*LDI* = 0), this new law arguably generalizes Fitts' law. Finally, we found unexpectedly high values for pause time (*PT*) and click time (*CT*), which we attribute to the cognitive load associated with the task at hand. We found a *linear* correlation between pause time (*PT*) and the length-distance index (*LDI*) and we often observed that the differences in pointing performance decreased as *LDI* increased. We also found that *CT* was strongly dependent on the nature of the target.

While this work is mainly theoretical, it has immediate practical applications. For example, many movements with high *LDI* correspond to users "shaking the mouse" to find the cursor, typically after a long pause, calling for more efficient ways of locating the cursor, especially for large screens and multiple-display settings. Reducing click time is also an interesting challenge as it does not seem to be linked to pure motor control. We intend to pursue this work by analysing other aspects of the data, by testing the combined index of difficulty in an experimental setting and by using our findings to guide the design of new interaction techniques.

REFERENCES

1. J. Accot and S. Zhai. Refining Fitts' law models for bivariate pointing. In *Proc. CHI '03*, pages 193–200. ACM Press, 2003.

2. T. Amer, A. Cockburn, R. Green, and G. Odgers. Evaluating swiftpoint as a mobile device for direct manipulation input. In *Proc. AUIC '07*, pages 63–70. Australian Comput. Soc., 2007.

3. R. Balakrishnan. "beating" fitts' law: virtual enhancements for pointing facilitation. *Inter. J. of Hum.-Comput. Stu.*, 61(6):857–874, 2004.

4. S. Card, T. P. Morn, and A. Newell. The keystroke-level model for user performance with interactive systems. *Comm. ACM*, 26:396–210, 1980.

5. O. Chapuis. Gestion des fenêtres : enregistrement et visualisation de l'interaction. In *Proc. IHM '05*, pages 255–258. ACM Press, 2005.

6. S. Coles. *An Introduction to Statistical Modeling of Extreme Values*. Springer-Verlag, 2001.

7. B. W. Epps. Comparison of six cursor control devices based on fitts' law models. In *Proc. Hum. Factors Soc.*, pages 327–331, 1986.

8. P. M. Fitts. The information capacity of the human motor system in controlling the amplitude of movement. *J. Exper. Psych.*, 47:381–391, 1954.

9. T. Grossman and R. Balakrishnan. Pointing at trivariate targets in 3d environments. In *Proc. CHI '04*, pages 447–454. ACM Press, 2004.

10. T. Grossman and R. Balakrishnan. A probabilistic approach to modeling two-dimensional pointing. *ACM Trans. Comput.-Hum. Interact.*, 12(3):435–459, 2005.

11. T. Grossman, N. Kong, and R. Balakrishnan. Modeling pointing at targets of arbitrary shapes. In *Proc. CHI '07*, pages 463–472. ACM Press, 2007.

12. Y. Guiard, F. Bourgeois, D. Mottet, and M. Beaudouin-Lafon. Beyond the 10-bit barrier: Fitts' law in multi-scale electronic worlds. In *Proc. IHM-HCI '01*, pages 573–587. Springer, 2001.

13. J. P. Hourcade, B. B. Bederson, A. Druin, and F. Guimbretière. Differences in pointing task performance between preschool children and adults using mice. *ACM Trans. Comput.-Hum. Interact.*, 11(4):357–386, 2004.

14. A. Hurst, S. E. Hudson, and J. Mankoff. Dynamic detection of novice vs. skilled use without a task model. In *Proc. CHI '07*, pages 271–280. ACM Press, 2007.

15. D. R. Hutchings, G. Smith, B. Meyers, M. Czerwinski, and G. Robertson. Display space usage and window management operation comparisons between single monitor and multiple monitor users. In *Proc. AVI '04*, pages 32–39. ACM Press, 2004.

16. E. Hutchins. *Cognition in the Wild*. MIT Press, 1987.

17. F. Hwang, S. Keates, P. Langdon, and J. Clarkson. Mouse movements of motion-impaired users: a submovement analysis. In *Proc. Assets '04*, pages 102–109. ACM Press, 2004.

18. ISO. Iso 9241-9:2000(e): Requirements for non-keyboard input devices (iso 9241-9). *Inter. Org. Stand.*, February 15, 2002.

19. M. Y. Ivory and M. A. Hearst. The state of the art in automating usability evaluation of user interfaces. *ACM Comput. Surv.*, 33(4):470–516, 2001.

20. W. E. Mackay and M. Beaudouin-Lafon. Diva: exploratory data analysis with multimedia streams. In *Proc. CHI '98*, pages 416–423. ACM Press, 1998.

21. I. S. MacKenzie. Fitts' law as a research and design tool in human-computer interaction. *Hum.-Comput. Interact.*, 7:91–139, 1992.

22. I. S. MacKenzie and W. Buxton. Extending Fitts' law to two-dimensional tasks. In *Proc. CHI '92*, pages 219–226. ACM Press, 1992.

23. I. S. MacKenzie and A. Oniszczak. A comparison of three selection techniques for touchpads. In *Proc. CHI '98*, pages 336–343. ACM Press/Addison-Wesley, 1998.

24. D. Meyer, J. Smith, S. Kornblum, R. Abrams, and C. Wright. Optimality in human motor performance: Ideal control of rapid aimed movements. *Psych. Review*, 95:340–370, 1988.

25. A. Sears and B. Shneiderman. Split menus: effectively using selection frequency to organize menus. *ACM Trans. Comput.-Hum. Interact.*, 1(1):27–51, 1994.

26. S. Seow. Information theoretic models of hci: A comparison of the hick-hyman law and fitts' law. *J. Hum.-Comput. Interact.*, 20(3):315–352, 2005.

27. S. Yun and G. Lee. Design and comparison of acceleration methods for touchpad. In *Proc. CHI '07*, pages 2801–2812. ACM Press, 2007.